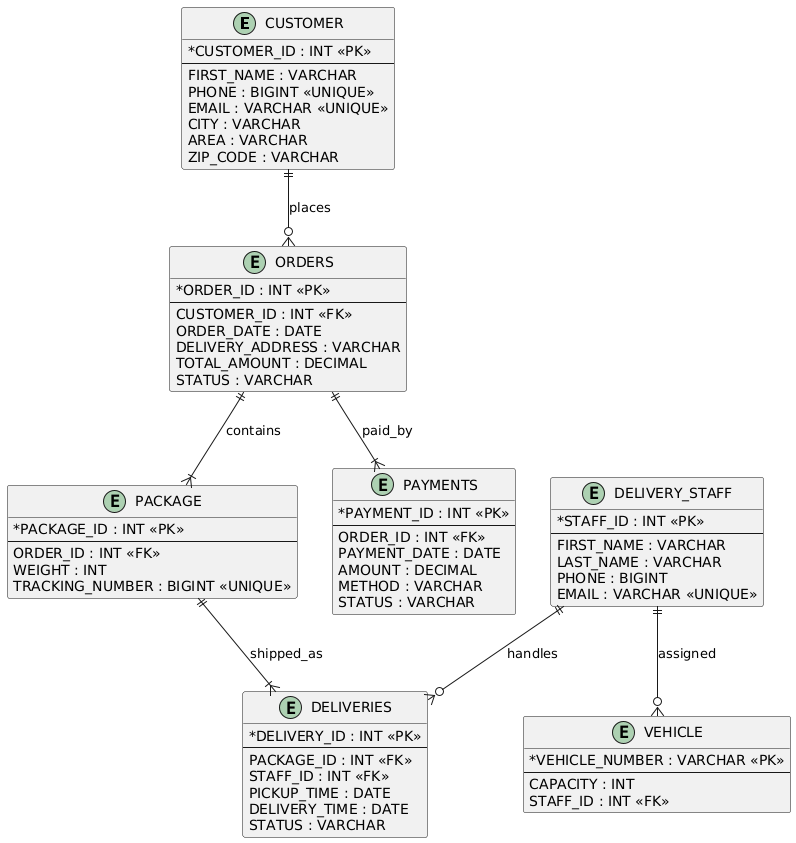
**Delivery Management System**

**Project Description:**

The Delivery Management System is designed to digitally manage customers, deliveries, orders, payment transactions, and staff allocation. The system aims to streamline parcel handling, payments, deliveries, and customer service. It focuses on efficient tracking, reporting, and real-time status updates.

**Entity Relationship Diagram:**



**Key Modules and Their Functions:**

**1. Customer Management:**

* Store customer details such as name, phone, email, city, area, and zip code.
* Track the cities from where customers place orders.
* Update customer information like city and personal details.

**2. Order Management:**

* Maintain order records with customer association, order date, delivery address, total amount, and status (Pending, Shipped, Delivered, Cancelled).
* Filter and retrieve orders based on amount ranges, status, or customers.
* Manage payment records linked to orders.

**3. Delivery Management:**

* Track package delivery details including pickup and delivery time, status (Transit, Delivered, Failed).
* Assign delivery staff to specific deliveries.
* Monitor the delivery status and update accordingly through triggers.

**4. Package Management:**

* Manage package records linked to orders.
* Store package weight and unique tracking numbers for each parcel.

**5. Staff Management:**

* Maintain delivery staff profiles including first name, last name, phone, and email.
* Assign delivery tasks and vehicles to specific staff members.

**6. Payment Management:**

* Record payments linked to orders with method (Cash, Card, UPI) and status (Paid, Failed, Returned).
* Calculate total sales and analyze payment trends.

**7. Vehicle Management:**

* Maintain delivery vehicle records including vehicle number, capacity, and assigned staff member.
* Link vehicles to the staff responsible for deliveries.

**1. Database Creation**

CREATE DATABASE PROJECTS\_learn;

GO

USE PROJECTS\_learn;

GO

**2. Table Creation with Constraints**

CREATE TABLE CUSTOMER (

CUSTOMER\_ID INT PRIMARY KEY,

FIRST\_NAME VARCHAR(50) NOT NULL,

PHONE BIGINT UNIQUE NOT NULL,

EMAIL VARCHAR(50) UNIQUE NOT NULL,

CITY VARCHAR(50),

AREA VARCHAR(100),

ZIP\_CODE VARCHAR(50)

);

CREATE TABLE DELIVERY\_STAFF (

STAFF\_ID INT PRIMARY KEY,

FIRST\_NAME VARCHAR(50) NOT NULL,

LAST\_NAME VARCHAR(50) NOT NULL,

PHONE BIGINT NOT NULL,

EMAIL VARCHAR(50) UNIQUE NOT NULL

);

CREATE TABLE ORDERS (

ORDER\_ID INT IDENTITY(1,1) PRIMARY KEY,

CUSTOMER\_ID INT,

ORDER\_DATE DATE,

DELIVERY\_ADDRESS VARCHAR(100),

TOTAL\_AMOUNT DECIMAL(10,3),

STATUS VARCHAR(50) CHECK (STATUS IN ('PENDING', 'SHIPPED', 'DELIVERED', 'CANCELLED')),

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID)

);

CREATE TABLE PACKAGE (

PACKAGE\_ID INT IDENTITY(1,1) PRIMARY KEY,

ORDER\_ID INT,

WEIGHT INT,

TRACKING\_NUMBER BIGINT UNIQUE,

FOREIGN KEY (ORDER\_ID) REFERENCES ORDERS(ORDER\_ID)

);

CREATE TABLE DELIVERIES (

DELIVERY\_ID INT IDENTITY(2,1) PRIMARY KEY,

PACKAGE\_ID INT,

STAFF\_ID INT,

PICKUP\_TIME DATE,

DELIVERY\_TIME DATE,

STATUS VARCHAR(20) CHECK (STATUS IN ('TRANSIT', 'DELIVERED', 'FAILED')),

FOREIGN KEY (PACKAGE\_ID) REFERENCES PACKAGE(PACKAGE\_ID),

FOREIGN KEY (STAFF\_ID) REFERENCES DELIVERY\_STAFF(STAFF\_ID)

);

CREATE TABLE PAYMENTS (

PAYMENT\_ID INT IDENTITY(3,1) PRIMARY KEY,

ORDER\_ID INT,

PAYMENT\_DATE DATE,

AMOUNT DECIMAL(10,3),

METHOD VARCHAR(15) CHECK (METHOD IN ('CASH', 'CARD', 'UPI')),

STATUS VARCHAR(15) CHECK (STATUS IN ('PAID', 'FAILED', 'RETURNED')),

FOREIGN KEY (ORDER\_ID) REFERENCES ORDERS(ORDER\_ID)

);

CREATE TABLE VEHICLE (

VEHICLE\_NUMBER VARCHAR(7) PRIMARY KEY,

CAPACITY INT,

STAFF\_ID INT,

FOREIGN KEY (STAFF\_ID) REFERENCES DELIVERY\_STAFF(STAFF\_ID)

);

GO

**3. Data Insertion (DML)**

**Insert Customers:**

INSERT INTO CUSTOMER (CUSTOMER\_ID, FIRST\_NAME, PHONE, EMAIL, CITY, AREA, ZIP\_CODE) VALUES

(1, 'Ahmed', 1234567, 'ahmed@gmail.com', 'Karachi', 'DHA Phase 6', '75500'),

(2, 'Ayesha', 2345678, 'ayesha@hotmail.com', 'Lahore', 'Gulberg III', '54000'),

(3, 'Usman', 3456789, 'usman@yahoo.com', 'Islamabad', 'F-10', '44000'),

(4, 'Zainab', 1122334, 'zainab@hotmail.com', 'Faisalabad', 'Peoples Colony', '38000'),

(5, 'Hassan', 4455667, 'hassan@gmail.com', 'Peshawar', 'University Town', '25000'),

(6, 'Maria', 5566778, 'maria@yahoo.com', 'Quetta', 'Satellite Town', '87300');

**Insert Delivery Staff:**

INSERT INTO DELIVERY\_STAFF (STAFF\_ID, FIRST\_NAME, LAST\_NAME, PHONE, EMAIL) VALUES

(1, 'Ali', 'Khan', 1112223, 'ali.khan@courier.pk'),

(2, 'Sara', 'Ahmed', 9998887, 'sara.ahmed@courier.pk'),

(3, 'Bilal', 'Yousaf', 3344556, 'bilal.yousaf@courier.pk'),

(4, 'Nadia', 'Khan', 7766554, 'nadia.khan@courier.pk');

**Insert Orders:**

INSERT INTO ORDERS (CUSTOMER\_ID, ORDER\_DATE, DELIVERY\_ADDRESS, TOTAL\_AMOUNT, STATUS) VALUES

(1, '2025-04-20', 'Street 12, DHA Phase 6, Karachi', 2500.500, 'DELIVERED'),

(2, '2025-04-21', 'House 7, Gulberg III, Lahore', 3200.000, 'SHIPPED'),

(3, '2025-04-22', 'House 25, G-11, Islamabad', 1800.750, 'DELIVERED'),

(4, '2025-04-23', 'Street 7, Peoples Colony, Faisalabad', 4600.000, 'PENDING'),

(5, '2025-04-23', 'Flat 8, University Town, Peshawar', 3100.250, 'SHIPPED'),

(6, '2025-04-24', 'Block C, Satellite Town, Quetta', 2200.900, 'CANCELLED');

### Insert Packages:

INSERT INTO PACKAGE (ORDER\_ID, WEIGHT, TRACKING\_NUMBER) VALUES

(1, 3, 987654321),

(2, 2, 876543210),

(3, 1, 112233445),

(4, 4, 223344556),

(5, 2, 334455667),

(6, 5, 445566778);

**Insert Deliveries:**

INSERT INTO DELIVERIES (PACKAGE\_ID, STAFF\_ID, PICKUP\_TIME, DELIVERY\_TIME, STATUS) VALUES

(1, 1, '2025-04-20', '2025-04-21', 'DELIVERED'),

(2, 2, '2025-04-21', NULL, 'TRANSIT'),

(3, 3, '2025-04-22', '2025-04-22', 'DELIVERED'),

(4, 4, '2025-04-23', NULL, 'TRANSIT'),

(5, 1, '2025-04-23', NULL, 'TRANSIT'),

(6, 2, '2025-04-24', NULL, 'FAILED');

### Insert Payments:

INSERT INTO PAYMENTS (ORDER\_ID, PAYMENT\_DATE, AMOUNT, METHOD, STATUS) VALUES

(1, '2025-04-21', 2500.500, 'CASH', 'PAID'),

(2, '2025-04-21', 3200.000, 'CARD', 'PAID'),

(3, '2025-04-22', 1800.750, 'CARD', 'PAID'),

(4, '2025-04-23', 4600.000, 'CASH', 'FAILED'),

(5, '2025-04-23', 3100.250, 'UPI', 'PAID'),

(6, '2025-04-24', 2200.900, 'CARD', 'RETURNED');

**Insert Vehicles:**

INSERT INTO VEHICLE (VEHICLE\_NUMBER, CAPACITY, STAFF\_ID) VALUES

('KHI1234', 500, 1),

('LHR5678', 600, 2),

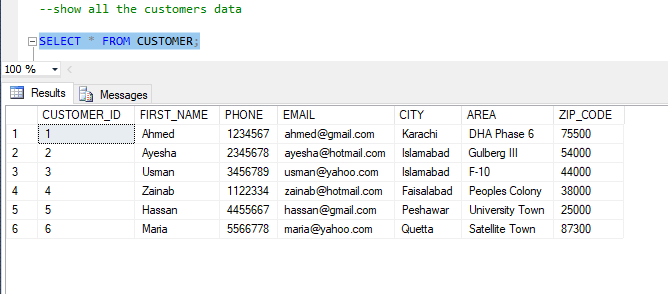
('ISB4321', 450, 3),

('FSD8765', 550, 4);

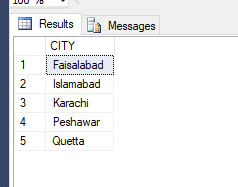
**4. SQL Queries (DML)**

**Select All Customers:**

SELECT \* FROM CUSTOMER;



**Select Distinct Cities:**



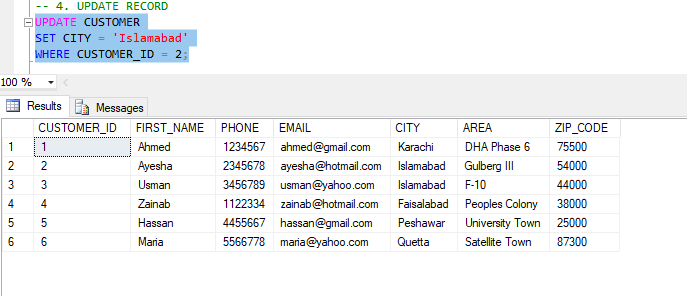
SELECT DISTINCT CITY FROM CUSTOMER;

**Update a Record:**

UPDATE CUSTOMER

SET CITY = 'Islamabad'

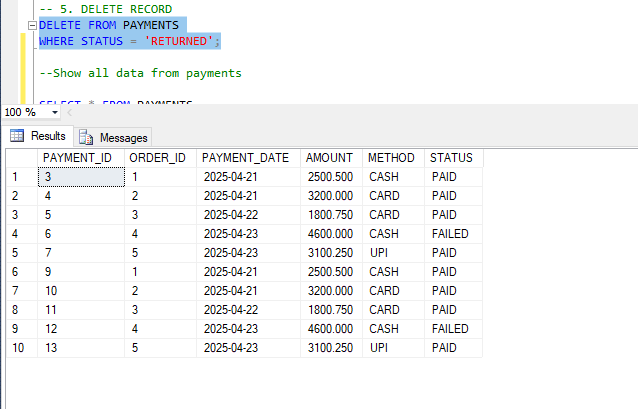
WHERE CUSTOMER\_ID = 2;



**Delete a Record:**

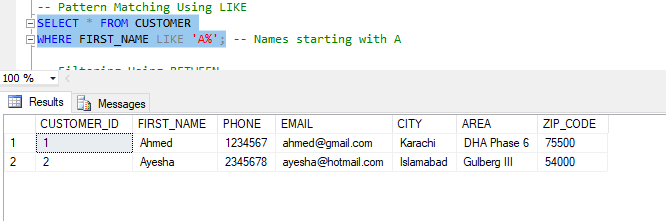
DELETE FROM PAYMENTS

WHERE STATUS = 'RETURNED';



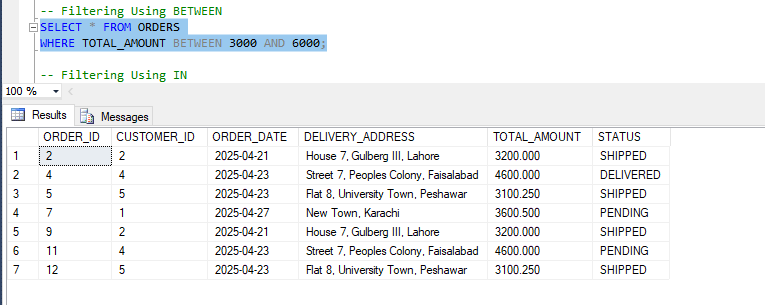
**Pattern Matching (LIKE):**

SELECT \* FROM CUSTOMER WHERE FIRST\_NAME LIKE 'A%';



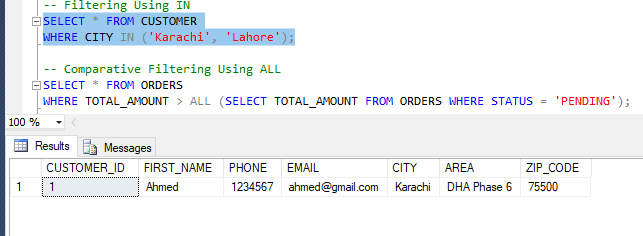
**BETWEEN Operator:**

SELECT \* FROM ORDERS WHERE TOTAL\_AMOUNT BETWEEN 3000 AND 6000;



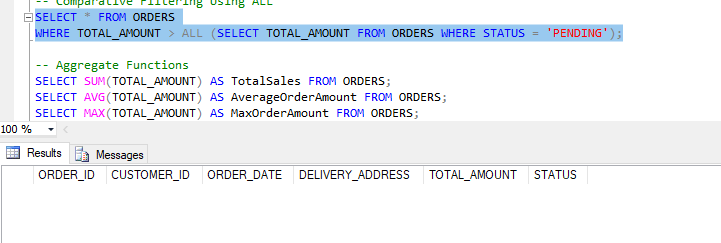
**Filtering (IN Operator):**

SELECT \* FROM CUSTOMER WHERE CITY IN ('Karachi', 'Lahore');



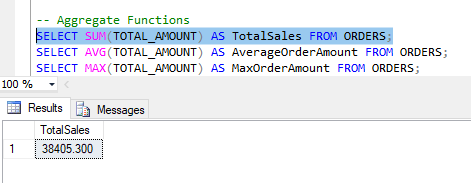
**Comparative Filtering (ALL):**

SELECT \* FROM ORDERS

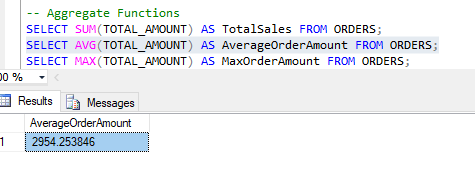
WHERE TOTAL\_AMOUNT > ALL (SELECT TOTAL\_AMOUNT FROM ORDERS WHERE STATUS = 'PENDING'); 

**Aggregate Functions:**

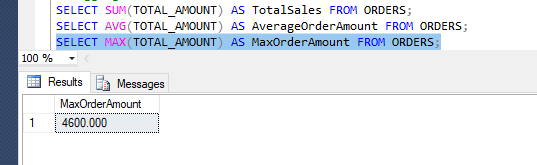
SELECT SUM(TOTAL\_AMOUNT) AS TotalSales FROM ORDERS;



SELECT AVG(TOTAL\_AMOUNT) AS AverageOrderAmount FROM ORDERS;



SELECT MAX(TOTAL\_AMOUNT) AS MaxOrderAmount FROM ORDERS;



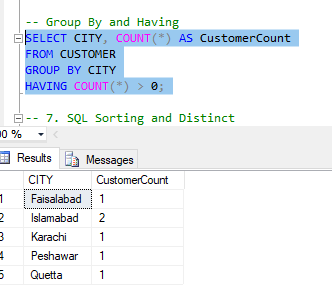
**Group By and Having:**

SELECT CITY, COUNT(\*) AS CustomerCount

FROM CUSTOMER

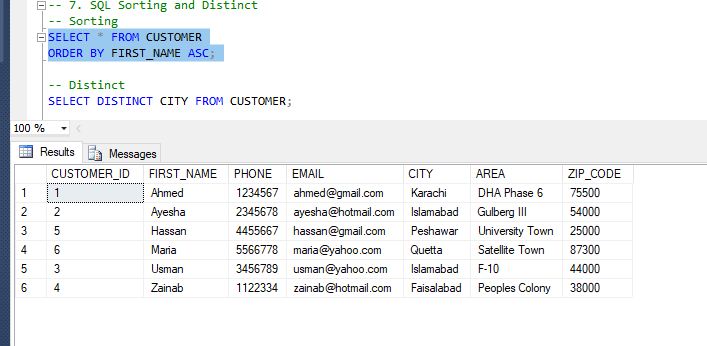
GROUP BY CITY

HAVING COUNT(\*) > 0;

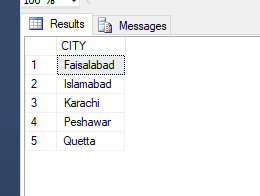


**Sorting and Distinct:**

SELECT \* FROM CUSTOMER ORDER BY FIRST\_NAME ASC;



SELECT DISTINCT CITY FROM CUSTOMER;



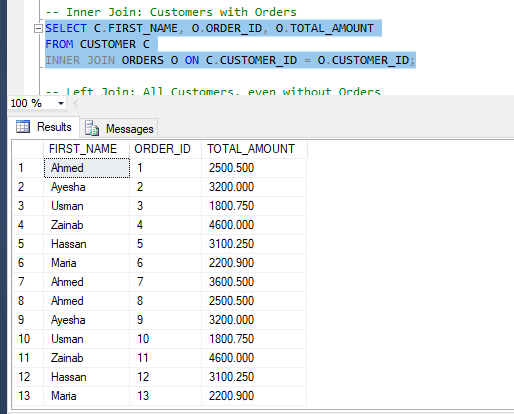
**5. SQL Joins**

**Inner Join:**

SELECT C.FIRST\_NAME, O.ORDER\_ID, O.TOTAL\_AMOUNT

FROM CUSTOMER C

INNER JOIN ORDERS O ON C.CUSTOMER\_ID = O.CUSTOMER\_ID;

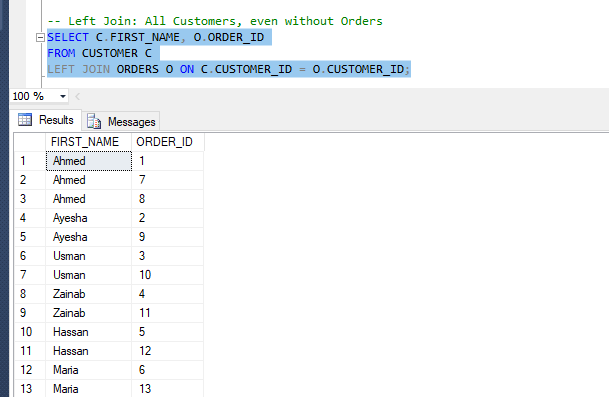


**Left Join:**

SELECT C.FIRST\_NAME, O.ORDER\_ID

FROM CUSTOMER C

LEFT JOIN ORDERS O ON C.CUSTOMER\_ID = O.CUSTOMER\_ID;

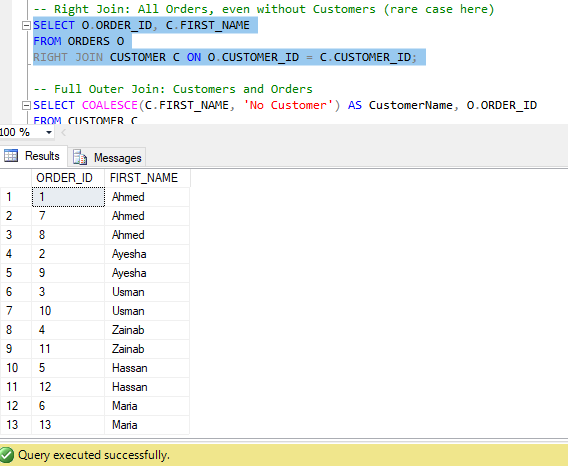


**Right Join:**

SELECT O.ORDER\_ID, C.FIRST\_NAME

FROM ORDERS O

RIGHT JOIN CUSTOMER C ON O.CUSTOMER\_ID = C.CUSTOMER\_ID;

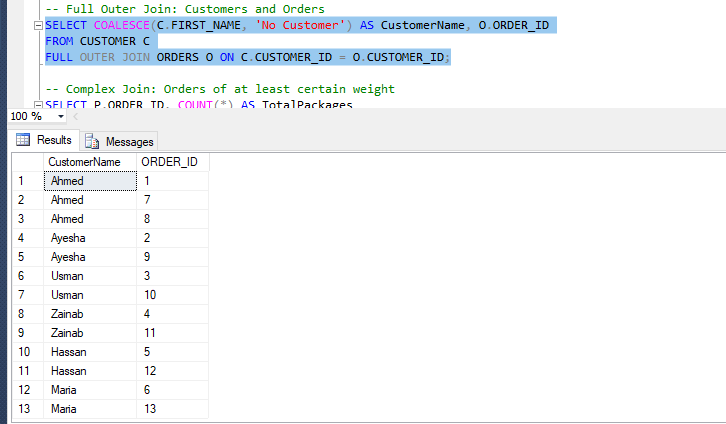


**Full Outer Join:**

SELECT COALESCE(C.FIRST\_NAME, 'No Customer') AS CustomerName, O.ORDER\_ID

FROM CUSTOMER C

FULL OUTER JOIN ORDERS O ON C.CUSTOMER\_ID = O.CUSTOMER\_ID;



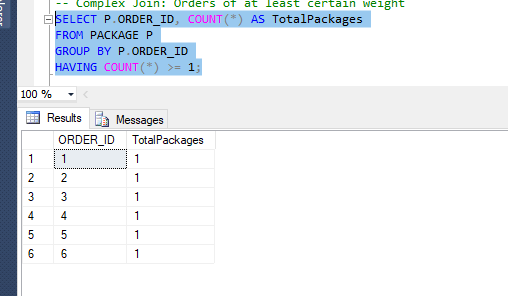
**Complex Join:**

SELECT P.ORDER\_ID, COUNT(\*) AS TotalPackages

FROM PACKAGE P

GROUP BY P.ORDER\_ID

HAVING COUNT(\*) >= 1;



**6. Stored Procedures**

**Retrieve Customers by City:**

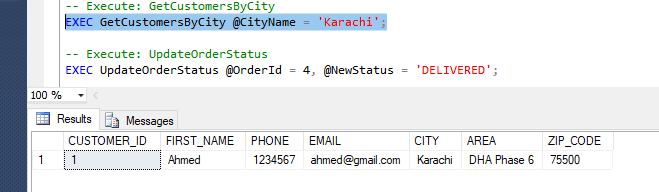
CREATE PROCEDURE GetCustomersByCity @CityName VARCHAR(50)

AS

BEGIN

SELECT \* FROM CUSTOMER WHERE CITY = @CityName;

END;



**Update Order Status:**

CREATE PROCEDURE UpdateOrderStatus @OrderId INT, @NewStatus VARCHAR(20)

AS

BEGIN

UPDATE ORDERS SET STATUS = @NewStatus WHERE ORDER\_ID = @OrderId;

END;

**Insert New Order:**

CREATE PROCEDURE InsertNewOrder

@CustomerId INT,

@OrderDate DATE,

@DeliveryAddress VARCHAR(100),

@TotalAmount DECIMAL(10,3),

@Status VARCHAR(50)

AS

BEGIN

INSERT INTO ORDERS (CUSTOMER\_ID, ORDER\_DATE, DELIVERY\_ADDRESS, TOTAL\_AMOUNT, STATUS)

VALUES (@CustomerId, @OrderDate, @DeliveryAddress, @TotalAmount, @Status);

END;

**Total Sales Calculation:**

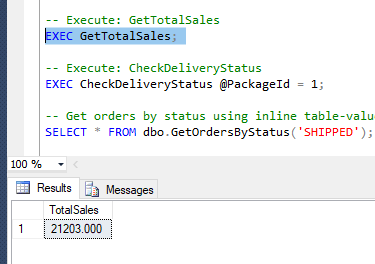
CREATE PROCEDURE GetTotalSales

AS

BEGIN

SELECT SUM(AMOUNT) AS TotalSales FROM PAYMENTS WHERE STATUS = 'PAID';

END;



**Delivery Status Check:**

CREATE PROCEDURE CheckDeliveryStatus @PackageId INT

AS

BEGIN

DECLARE @status VARCHAR(20);

SELECT @status = STATUS FROM DELIVERIES WHERE PACKAGE\_ID = @PackageId;

IF (@status = 'DELIVERED')

PRINT 'Package Delivered Successfully!';

ELSE

PRINT 'Package Not Delivered Yet.';

END;

**7. SQL Functions and Views**

**Inline Table-Valued Function:**

CREATE FUNCTION dbo.GetOrdersByStatus(@Status VARCHAR(50))

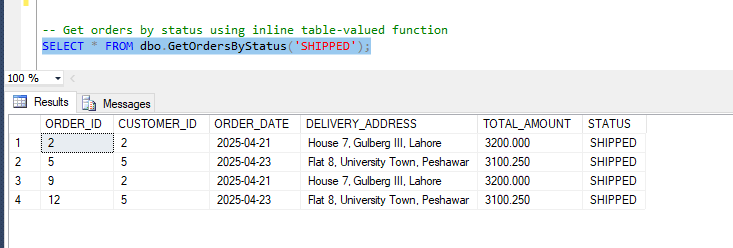
RETURNS TABLE

AS

RETURN (

SELECT \* FROM ORDERS WHERE STATUS = @Status

);



**Multi-Statement Table-Valued Function:**

CREATE FUNCTION dbo.CustomerOrderSummary()

RETURNS @Summary TABLE

(

CustomerName VARCHAR(50),

TotalOrders INT

)

AS

BEGIN

INSERT INTO @Summary

SELECT C.FIRST\_NAME, COUNT(O.ORDER\_ID)

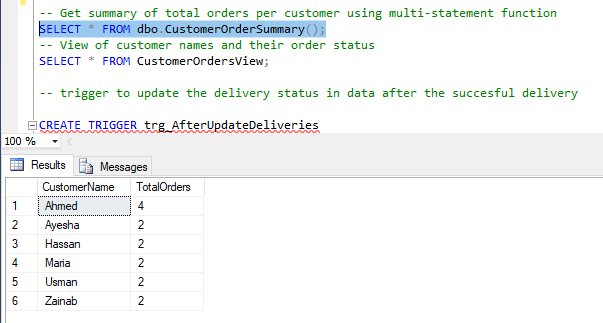
FROM CUSTOMER C

LEFT JOIN ORDERS O ON C.CUSTOMER\_ID = O.CUSTOMER\_ID

GROUP BY C.FIRST\_NAME;

RETURN;

END;



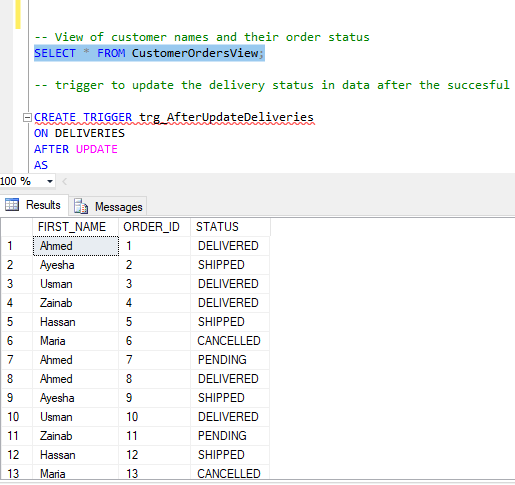
**View Creation:**

CREATE VIEW CustomerOrdersView AS

SELECT C.FIRST\_NAME, O.ORDER\_ID, O.STATUS

FROM CUSTOMER C

JOIN ORDERS O ON C.CUSTOMER\_ID = O.CUSTOMER\_ID;



**8. Trigger Creation**

**Update Trigger for Deliveries:**

CREATE TRIGGER trg\_AfterUpdateDeliveries

ON DELIVERIES

AFTER UPDATE

AS

BEGIN

PRINT 'Delivery record updated!';

END;

**Trigger Execution:**

UPDATE DELIVERIES

SET DELIVERY\_TIME = GETDATE()

WHERE DELIVERY\_ID = 1;